BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

Vista

Last updated: Thu Aug 22 16:53:37 MET DST 1996 by Oscar Nierstrasz

Summary

Vista is a generic visual composition tool developed by the Object Systems Group at the University of Geneva as part of ITHACA, an Esprit technology integration project whose goal was to produce a complete, object-oriented application development environment. Vista differs from other visual composition tools in that it is domain-neutral: Vista can be adapted to different component sets by specifying what kinds interfaces (*ports* and *links*) are supported by components, what the compatibility rules are, how components are glued together, and how components are mapped to their visual presentations.

Vista is described in detail in the PhD dissertation of Vicki de Mey.

Component Model

Component Definition

A component consists of a *behaviour* and a *presentation* (i.e., a `model' and a `view'). The behaviour implements the services of the software components, and the presentation provides a visual interface for Vista. Each component is parameterized by a number of *input* and *output ports* that stand, respectively, for required and provided services. Applications are specified by linking together compatible ports of different components. The visual presentation of the ports (and of the links) may vary from one component set to another, just as the presentations of the components themselves may vary.

Interface

The interface to a component consists purely of input and output ports.

An input port of one component may be linked to the output port of another component (signifying that a required service of the first component is provided by the second) if and only if the two ports are *compatible*. The provider of a component set specifies what types of ports may be supported by the components and what the compatibility rules are. A simple form of subtyping is supported, so an input port of type A that requires an output port of type B may be satisfied by an output port of type B'.

Types are uninterpreted. Ports may stand for completely different kinds of services in different component sets. For example, ports may represent functions and function parameters for some components, and they may stand for input and output streams for another component set.

Genericity is not directly supported by Vista, but since ports can represent arbitrary values, in principle one could define a component set in which ports represents component types.

Encapsulation Boundary

Components are weakly encapsulated. Vista is only interested in the ports provided by a component, and the distinction between behaviour and presentation. A component is free to make use of any other interfaces known or not known by Vista in order to implement its behaviour. From Vista's point of view, components are black-box entities, but from the point of view of an implementor of components, they are white-box.

Granularity

Each component is realized by a set of C++ objects, representing the behaviour, the presentation, and the ports. The component itself may be of arbitrary granularity, however (i.e., as small as a single data value, or as large as a remote application).

Generality

Vista's type model being uninterpreted, nothing prevents one from developing a component set in which components can be passed as values at run-time. Vista, however, would not necessarily know about those components or links between them.

Vista supports construction of new components by encapsulation of compositions as components. A set of linked components can be encapsulated by assigning a presentation to the composition, and by specifying which ports of the constituent components should be visible at the boundary of the new component.

Binding Technology

Components are composed by linking input and output ports. Since ports are uninterpreted, the provider of a component set must also provide implementations of the abstract port and link classes, which realize the `glue' between components. The nature of the glue is thus deferred to the component framework itself. Component providers are free to implement any kind of binding mechanisms whatsoever.

A composition is essentially a static graph, which is built by a Vista user during an editing session. Vista checks whether links that the user attempts to establish are allowed by the compatibility rules specified for the component set, and instantiates link objects to glue ports together.

Compositions can be saved and restored, but they are neither compiled nor interpreted. Since a composition is a collection of connected objects, the composition itself must decide what it should do. It may be a running application itself, or it may consist of components that know how to generate code that is then compiled by a separate tool. Both of these approaches have been used in experimental applications of Vista.

Semantics

There are no formal semantics for the behaviour of Vista components. Vista compositions are (tripartite) graphs of components, input and output ports. The type model allows one to specify some simple compatibility rules between types of ports. No analysis tools are provided, though it is clear that this would be needed for a production-quality tool.

[Top | Summary | Component Model | Usage | Discussion | References]

Usage

[Domain | Architecture | Specification | Interoperability | Process]

Application Domain

Vista is intended to be domain-independent. It has been applied to:

1. workflow specification,
2. multimedia composition, and
3. requirements specification.

Architecture

Vista can be adapted to different component sets. Each component set supports a different generic architecture, which is specified by the kinds of input and output ports provided and their compatibility rules.

A generic architecture, then, is essentially a specification of the kinds of graphs (or compositions) that can be constructed.

Specification Medium

In principle, components can be implemented in any programming language, as long as a C interface is provided. A very simple language is used for specifying port types and compatibility rules.

In order to adapt a component set to Vista, the component provider must implement the C++ concrete classes that realize ports, links, behaviours and components. (These are `glue' objects that provide the interface between Vista and the component set itself.)

Interoperability and Distribution

Heterogeneous components can be supported by wrapping them in C++ objects.

Vista provides no support for distribution, though components are permitted to make use of the network (as is the case in the multimedia application).

Components must be compiled and linked together with Vista and the interface code. Compositions are saved in files as graph representations. Only components that have been built as compositions can be saved in this way.

Software Process

Vista was developed as part of an integrated environment, including a Software Information Base (SIB) and a Requirements Collection and Specification Tool (RECAST). Vista is intended to be used mainly during implementation and maintenance. RECAST, however, which is used in the early phases of the software process, was actually prototyped using Vista, by viewing requirements specifications as components. Querying and navigation is provided by the SIB.

[Top | Summary | Component Model | Usage | Discussion | References]

Discussion

Vista is unusual in that virtually all existing visual composition tools are domain-specific (i.e., they can *only* be used for user interface building, or for workflow design, etc.), whereas Vista can be adapted to arbitrary domains. As a proof-of-concept prototype, Vista demonstrates that a generic visual composition tool is feasible.

Some of the open problems are:

- *Presentation:* It is hard to cope well with complex compositions. Different ways of viewing components, compositions, ports and links, must be provided, with convenient ways of switching between views. Interaction needs to be customizable.
- *Semantics:* The underlying graph model of Vista is rather weak. It should be possible to specify richer composition models so that one can reason about more complex forms of compatibility, or permit or prohibit cycles (for example).

[Top | Summary | Component Model | Usage | Discussion | References]

References

- Vicki de Mey's PhD thesis
- Visual Composition of Software Applications
- Various papers

[Top | Summary | Component Model | Usage | Discussion | References]

[Front Page | Index]